Sheet 6 (2D Graphics)

1.

a) **glClear(GL_COLOR_BUFFER_BIT);** This function clears the drawing window. Clearing is done by painting the drawing window using the color specified by the function **glClearColor*****. The frame buffer that will be cleared is the color buffer. Other frame buffers like the DEPTH buffer, when used, are not affected by this function call.

b) **glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);** This function initializes the OpenGL graphics library by specifying *RGBA* color model rather than index color model. It also specifies using a single frame buffer rather than double-buffered window (two frame buffers). It also specifies that the graphics window has an associated depth frame buffer.

c) **glEnable(GL_DEPTH_TEST);** This function enables the simple hidden surfaces removal algorithm that hides the surfaces based on its z location. That is if two pints in the scene have the same x and y projections on the clipping window, the one that has a smaller z is rendered by the system while the other is removed.

d) **glOrtho(-50. 0,50.0 ,-50.0, 50.0, -50.0, 50.0);** This function specifies the view volume in the world coordinates that will be imaged or projected. This volume is a parallelogram that extends from -50.0 to 50.0 in the x direction, from -50.0 to 50.0 in the y direction and from -50.0 to 50 in the z direction. The function also specifies parallel projection which means that all the contents of the volume described will be projected on a projection plane parallel to the x y plane. This function is usually used when drawing 2D graphics using OpenGL. In the case the objects are usually specified in the x y plane.

2. We can output text using the OpenGL graphic library in one of two main ways: using **stroke** text and using **raster** text. Using stroke text, each character is drawn as an object that has a geometrical shape. The character as an object is specified in terms of the OpenGL primitives that pass inside the entire graphic pipeline. On the other hand, when using raster text, Characters are defined as rectangles of bits called bit blocks. Each block defines a single character by the pattern of 0 and 1 bits in the block. A raster character can be placed in the frame buffer rapidly by a Bit-block-transfer operation, which moves the block of bits using a single function call.

	Stroke text	Raster text
Flexibility	We have a high degree of flexibility when writing stroke text. Since characters are geometric objects, we can apply all the capabilities of the GL system in drawing them, transforming them to obtain rotated/inverted… text, shade them the way we wish, etc.	There is a fixed pattern for each character. The only thing we can do on this pattern is to scale it using pixel replication

Processing time	Since stroke text passes by the entire graphics pipelines, it requires time and computational resources to render	Raster text is simple and fast, we can write the character blocks in the frame buffer directly.

3.